UNIT-IV

Trees

Basic Tree Concepts :-

Tree: A tree is a non-linear data structure, which is mainly used to store data i.e., hierarchical in nature.

Trees are used extensively in computer science to represent algebraic formulas, as an efficient method for searching large, dynamic lists and for such diverse applications as artificial intelligence systems and encoding algorithms.

Definition: A tree is defined as a finite set of one or more nodes such that

(i) there is a specially designated node called the root and

(ii) the rest of the nodes could be partitioned into t disjoint sets ($t \geq 0$) each set representing a tree T_i, $i = 1, 2, \ldots t$ known as subtree of the tree.

ex:

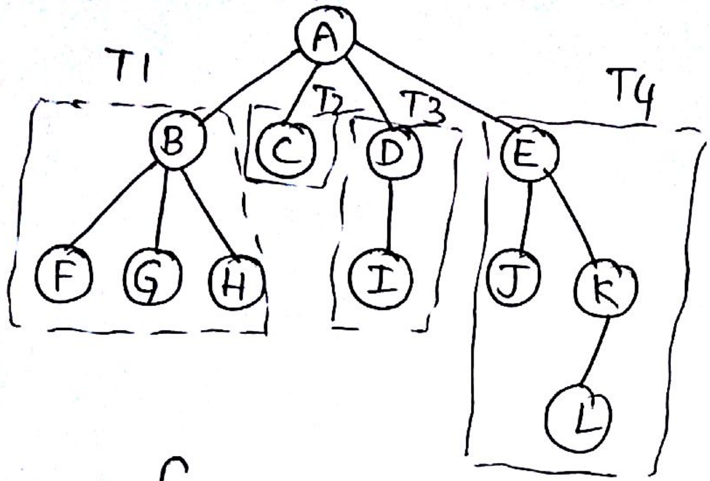


fig: Tree

T_1, T_2, T_3, T_4 are the subtrees.

→ The definition of the tree emphasizes on the aspect of

(i) connectedness

(ii) absence of closed loops & what are termed cycles.

(i) Connectedness: Beginning from the root node, the structure of the tree permits connectivity of the root to every other node in the tree. In general, any node is reachable from any where in the tree.

(ii) absence of cycles: With branches, providing the links between the nodes, the structure ensures that no set of nodes link together to form a closed loop or a cycle.

Basic Terminologies:

1. **NODE**: A node of the tree represents an item of information.

2. **Branch**: The links b/n the nodes termed as branches, represent an association b/n the items of information.

3. **Root node**: The root node is the topmost node in the tree i.e.; a node which has no parent.

4. **Parent node**: The parent of a node is the immediate predecessor of a node

(or)

A node having further subtrees

(or)

A node is a parent if it has successor nodes.

5. **Child node**: All the immediate successors of a node are known as child nodes.

(or)

A node with a predecessor is called a child node.

6. **Sibling nodes**: Two or more nodes with the same parent are siblings.

7. **Leaf node**: The node which is at the end and does not have any child is called leaf. It is also termed as "terminal node". or "external node".

8. Ancestor node : An ancestor is any node in the path from the root to the node

9. Descendent node: A descendent is any node in the path below the parent node, i.e, all nodes in the path from a given node to a leaf are descendents of that node.

10. Internal node : A node that is not a root or a leaf is known as an internal node because it is found in the middle portion of a tree. These are also called as "non-terminal nodes".

ex:

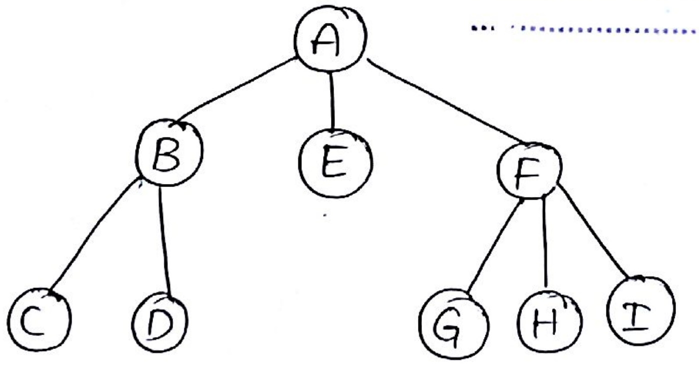


fig: Tree

1. Nodes : A, B, C, D, E, F, G, H, I
2. Branches : AB, AE, AF, BC, BD, FG, FH, FI
3. Root : A
4. Parents : A, B, F
5. child nodes : B, E, F, C, D, G, H, I

6. Siblings : {B, E, F}, {C, D}, {G, H, I}

7. leaf nodes : C, D, E, G, H, I

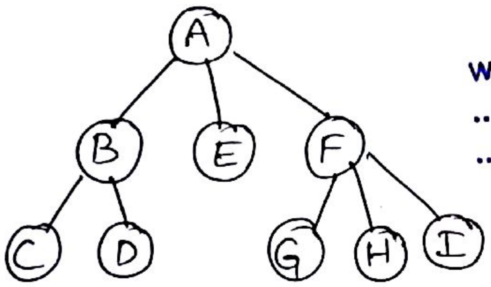
8. Ancestor nodes : for H :- {A, F}
for C :- {A, B}

9. Descendent nodes : for A :- {F, G}, {F, H}, {F, I} {E}, {B, C}, {B, D}

10. Internal nodes : {B, F}

11. <u>Degree</u> : For undirected tree, the no. of branches associated with a node is the degree of the node.

<u>ex</u>:



degree for A = 3
B = 3
C = 1
D = 1

degree for E = 1
F = 4
G = 1
H = 1
I = 1

12. <u>Indegree</u> : In directed tree, when the branch is directed toward the node, it is known as indegree.

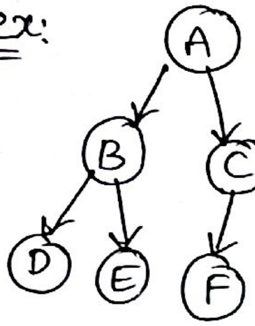
(or)

It is the no. of edges arriving at a node.

→ The indegree of the root is <u>zero</u>.

→ With the exception of the root, all of the nodes in a tree must have an indegree of <u>exactly one</u>, i.e., they may have only one predecessor.

<u>ex:</u>

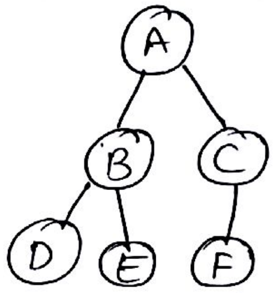


<u>fig</u>: directed tree

in-degree for A = 0

" for B, C, D, E, F = 1

<u>ex:</u>



<u>fig</u>: Undirected tree

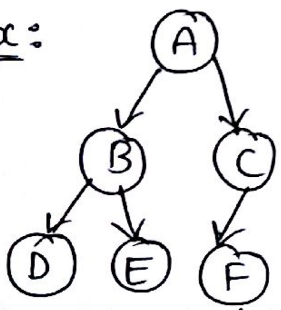
in-degree for A = 0

for B, C, D, E, F = 1

13. <u>Outdegree</u>: When the branch is directed away from the node, it is known as outdegree.

→ For a leaf node, the outdegree is <u>zero</u>.

<u>ex:</u>



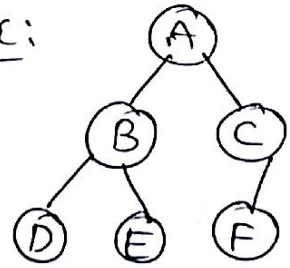
<u>fig</u>: Directed Tree

outdegree for A, B = 2

for C = 1

for D, E, F = 0

<u>ex:</u>



<u>fig</u>: Undirected tree

outdegree for A, B = 2

for C = 1

for D, E, F = 0

The sum of indegree and outdegree is the degree of the node.

14. **Path:** A path is a sequence of nodes in which each node is adjacent to the next one.

ex:

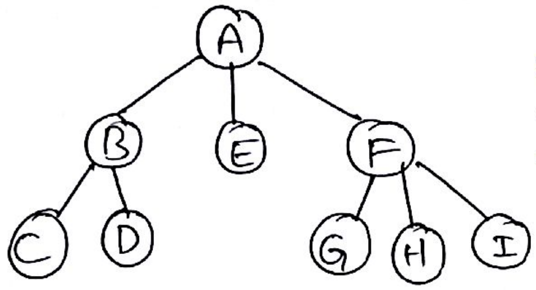


fig: Tree

The path from root to node 'H' is A, F, H.

15. **Level:** The level of a node is its distance from the root.

→ Because the root has a zero distance from itself the root is at level '0'.

→ The children of the root are at level 1, their children are at level 2., & so forth.

ex:

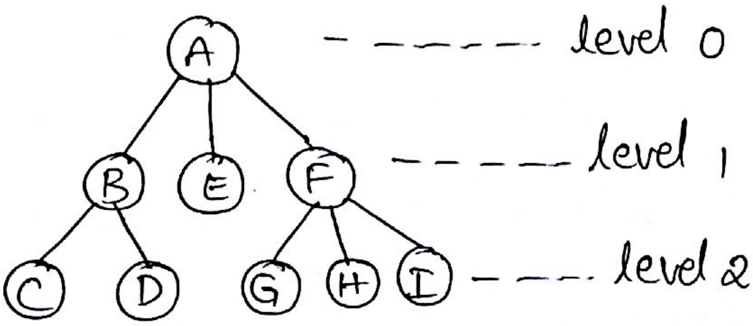


fig: Tree.

16. **Height** :- The height of a tree is the level of the leaf in the longest path from the root plus 1.

(or)

It is the total no. of nodes on the path from the root node to the deepest node in the tree.

→ The height of an empty tree is -1.

→ A tree with only a root node has a height of 1.

ex:

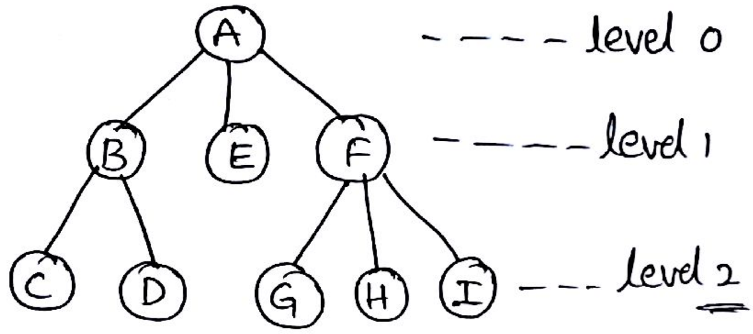


fig: tree

∴ height = 2 + 1 = 3.

17. **Depth** :- Depth of a node to be the length of the longest path from the root node to that node.

→ The depth of the root node is zero.

ex:

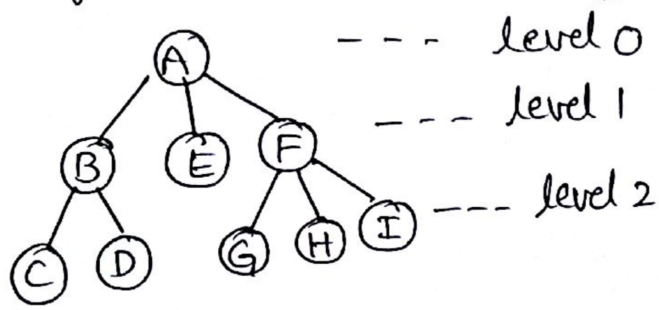


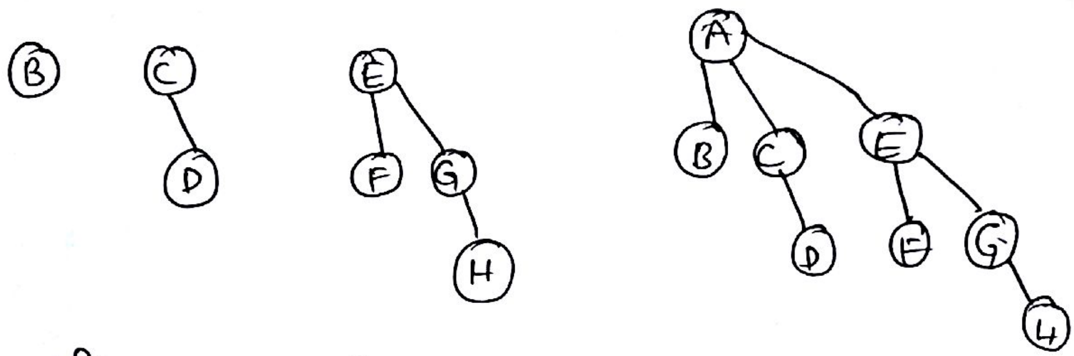
fig: tree

∴ depth = 2.

$$\boxed{\therefore \text{depth} = \text{height} - 1}$$

18. <u>Forest</u> :- A forest is a disjoint union of trees. A set of disjoint trees (or forest) is obtained <u>by deleting the root</u> and the edges connecting the root node to nodes at level 1.

<u>ex</u>:

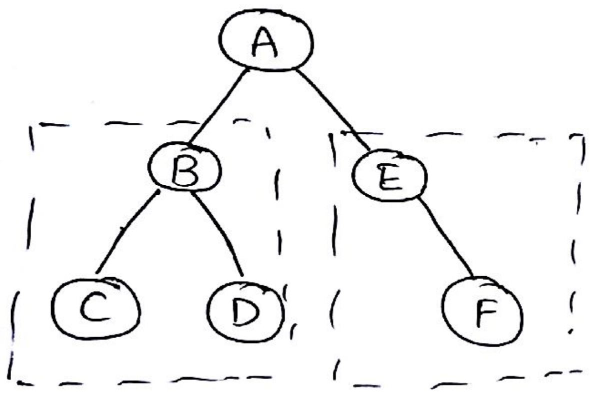


<u>fig</u>: ~~From~~ forest & its corresponding trees.

Binary Trees :- A binary tree is a tree in which no node can have more than 2 subtrees, the maximum outdegree for a node is 2.

→ A node can have zero, one or two subtrees. These subtrees are designated as the left subtree and the right subtree.

ex:



Left Subtree   Right Subtree

→ A null tree is a tree with no nodes.

Properties :- There are several properties for binary trees that distinguish them from general trees.

1. **Height of Binary trees**: The height of binary trees can be mathematically predicted.

2. **Maximum Height**: The maximum height, H_{max}, is

$$H_{max} = N$$

ex: Given three nodes to be stored in a binary tree, What is the maximum height?

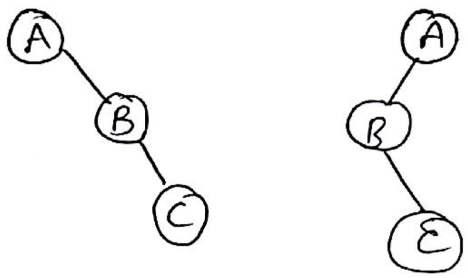
Sol: $N = 3$.

$\therefore$ The max. height is 3.

→ A tree with a max. height is rare.

→ It occurs when all of the nodes in the entire tree have only one successor.

ex:



3. Minimum Height:- The minimum height of the tree is,

$$\boxed{H_{min} = \left[\lfloor \log_2 N \rfloor\right] + 1}$$

ex: Given three nodes to be stored in a binary tree, what is the min. height?

Sd: $N = 3$

$$\therefore H_{min} = \left[\lfloor \log_2 3 \rfloor\right] + 1 = 2$$

4. Minimum Nodes:- Given a height of the ~~tr~~ binary tree, H, the min. of nodes in a tree ~~of specified~~ ~~ht height, H~~, ~~the m~~ are given as,

$$\boxed{N_{min} = H}$$

Scanned by CamScanner

ex: Given a tree of height 3, what is the min no. of nodes that can be stored?

sol:

$$N_{min} = H$$

$$\therefore N_{min} = 3.$$

5. Maximum Nodes: Each node can have only two descendents. Given a height of the binary tree, H, the max. no. of nodes in the tree is given as,

$$\boxed{N_{max} = 2^H - 1}$$

ex: Given a tree of height 3, what is the max. no. of nodes that can be stored?

sol: $H = 3$

$$\therefore N_{max} = 2^H - 1 = 2^3 - 1 = 8 - 1 = 7.$$

6. In any binary tree, the max. no. of nodes on level l is 2^l, where $l \geq 0$.

7. For any non-empty binary tree, if n is the no. of nodes & e is the no. of edges, then $\boxed{n = e + 1}$

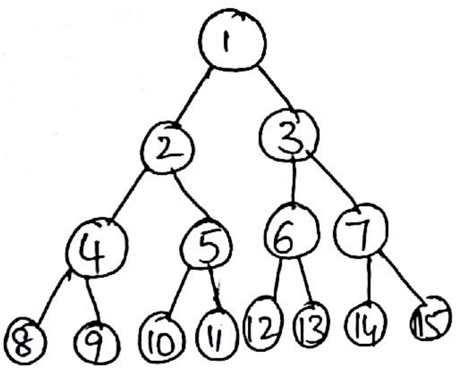
8. For any non-empty binary tree, if n_0 is the no. of leaf nodes (degree = 0) & n_2 is the no. of internal

nodes (degree = 2), then $\boxed{n_0 = n_2 + 1}$

9. The total no. of binary trees possible with n nodes is, $\boxed{\frac{1}{n+1} \, {}^{2n}C_n}$

10. <u>Full Binary Tree</u> :- A binary tree is a full binary tree if it contains the maximum possible number of nodes ~~all~~ at all levels.

<u>ex</u>:



max.no.of nodes

---- level 0 $2^0 = 1$

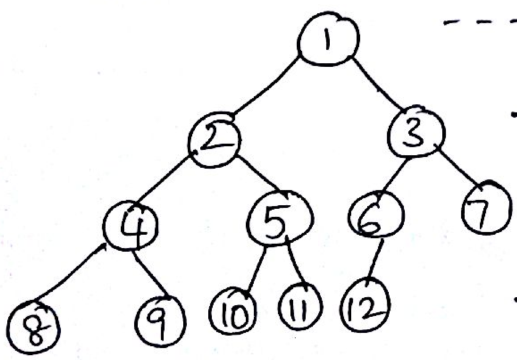
---- level 1 $2^1 = 2$

---- level 2 $2^2 = 4$

---- level 3 $2^3 = 8$

11. <u>Complete Binary Tree</u> :- A binary tree is said to be a complete binary tree, if all its levels, except possibly the last level, have the maximum number of possible nodes.

ex:



max. no. of nodes

---- level 0 $2^0 = 1$

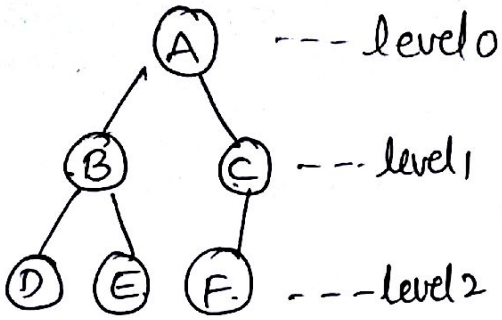
---- level 1 $2^1 = 2$

---- level 2 $2^2 = 4$

---- level 3 (last level) $\boxed{5}$ (exception)

12. The height of a complete binary tree with n no. of nodes is, $\lceil \log_2 (n+1) \rceil$.

ex:



$n = 6$

$\text{height} = \lceil \log_2 (n+1) \rceil$

$= \lceil \log_2 7 \rceil = \underline{3}$

Fig: Complete Binary tree

Representation of Binary tree:- A binary tree must represent a hierarchical relationship between a parent node and at most 2 child nodes.

→ There are 2 methods used for representing binary trees.

1. Linear or sequential Representation using arrays.
2. Linked Representation using pointers.

1. Linear Representation:- Sequential representation of trees is done using single or one-dimensional arrays. This type of representation is static i.e., a block of memory for an array is to be allocated before going to store the actual tree in it, once the memory is allocated, the size of the tree will be restricted as the memory permits.